* Add liquidity function is used to add liquidity in the contract. It is a function in clearing house contract.

It takes 1 input parameter which is called (AddLiquidityParams calldata params) which is a type of struct in IclearingHouse.

struct AddLiquidityParams {

address baseToken;

uint256 base;

uint256 quote;

int24 lowerTick;

int24 upperTick;

uint256 minBase;

uint256 minQuote;

bool useTakerBalance;

uint256 deadline;

}

* Where –
* baseToken: the address of the base token; specifies which market you want to trade in
* isBaseToQuote: true for shorting the base token asset, false for longing the base token asset
* isExactInput: for specifying exactInput or exactOutput ; similar to UniSwap V2's specs
* amount: the amount specified. Depending on the isExactInput parameter, this can be either the input amount or output amount.
* oppositeAmountBound: the restriction on how many token to receive/pay, depending on isBaseToQuote & isExactInput
  + isBaseToQuote && isExactInput: want more output quote as possible, so we set a lower bound of output quote
  + isBaseToQuote && !isExactInput: want less input base as possible, so we set a upper bound of input base
  + !isBaseToQuote && isExactInput: want more output base as possible, so we set a lower bound of output base
  + !isBaseToQuote && !isExactInput: want less input quote as possible, so we set a upper bound of input quote
* deadline: the restriction on when this tx should be executed; otherwise, it fails
* sqrtPriceLimitX96: the restriction on the ending price after the swap. 0 for no limit. This is the same as sqrtPriceLimitX96 in the UniSwap V3 contract.
* referralCode: the referral code for partners

Returns:

* deltaBase: the amount of base token exchanged
* deltaQuote: the amount of quote token exchanged
* \_checkMarketOpen – This is a function in clearinghouse contract which uses check token market is open or not. This uses Ibasetoken contrat which uses enum named status to check status of token market.
* IAccountBalance(\_accountBalance).registerBaseToken(trader, params.baseToken); - Here we use accountbalance contract to check the the token is registered to the user or not. In this accountbalance contract they have a mapping of user -> addressoftoken[] . means user are mapped to array of addresses of tokens. So if the token is present they return other they add token in it.
* Funding.Growth memory fundingGrowthGlobal = \_settleFunding(trader, params.baseToken); - This is a function of clearinghouse contract in which Settles trader's funding payment to his/her realized pnl. This settlefunding function calls settlefunding function of exchange contract which works as –
  + - require(IMarketRegistry(\_marketRegistry).hasPool(baseToken), "EX\_BTNE"); - checks in the market registry contract that the mapping poolMap have this token using haspool function.
    - (fundingGrowthGlobal, markTwap, indexTwap) = \_getFundingGrowthGlobalAndTwaps(baseToken); this calls another function of exchange contract \_getFundingGrowthGlobalAndTwaps.
      * bool marketOpen = IBaseToken(baseToken).isOpen(); - use base token contract to check market is open or not. Checks enum status
      * uint256 timestamp = marketOpen ? \_blockTimestamp() : IBaseToken(baseToken).getPausedTimestamp(); - to check the last time when last market was open or the current time using bas token contract.
      * if (\_firstTradedTimestampMap[baseToken] != 0) - takes this pmapping of address -> uint256 from exchange storage contract and checks if it is not zero for base token.
      * twapInterval = IClearingHouseConfig(\_clearingHouseConfig).getTwapInterval(); - gets twap interval from clearhouseconfig contract. It has a function which returns the \_twapInterval variable.
      * uint32 deltaTimestamp = timestamp.sub(\_firstTradedTimestampMap[baseToken]).toUint32(); - Here it differentiates the timesatamp and firsttradedtimstamp and then convert it into uint32 using toint32 function from safecast contract.
      * twapInterval = twapInterval > deltaTimestamp ? deltaTimestamp : twapInterval – Here twap interval value is assigned , the smaller value of the two is assigned to it.